EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	115807	(application with server)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:27
L2	188	(application with server) same (redundant with database)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:28
L3	2	(application with server) same (redundant with database with (modification or mofif$6))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:29
L4	295	(application with server) same (database with (modification or mofif$6))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:29
L5	13	(application with server) same (database with (modification or mofif$6)) and (commit$6 with command)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:30
L6	6	(application with server) same (database with (modification or mofif$6)) and (commit$6 with command) and cache	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:30
L7	0	(application with server) same (database with (modification or mofif$6)) and (commit$6 with command) and cache and synchronization	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:31
L8	0	(application with server) same (database with (modification or mofif$6)) and (commit$6 with command) and synchronization	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:32
L9	43	(database with (modification or mofif$6)) and (commit$6 with command) and synchronization	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:32

EAST Search History

L10	1	(database with (modification or mofif$6)) and (commit$6 with command) and synchronization with cache	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 19:34
L11	3407300	(commit$4 or (database$1 or (data near3 base) or db$1))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 20:01
L12	4148	(commit$4 with (database$1 or (data near3 base) or db$1))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 20:02
L13	37	(commit$4 with (database$1 or (data near3 base) or db$1)) and (synchroniz$6 with cluster$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 20:03
L14	37	(commit$4 with (database$1 or (data near3 base) or db$1)) and (synchroniz$6 with cluster$4) and (modif$6 or modificat$5 or delet$4 or add$4 or insert$4 or updat$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 20:09
L15	18	(commit$4 with (database$1 or (data near3 base) or db$1)) and (synchroniz$6 with cluster$4) and (modif$6 or modificat$5 or delet$4 or add$4 or insert$4 or updat$4) and "707"/$.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 20:14
L16	19	14 not 15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/04 20:15

PALM INTRANET

Day : Saturday
Date: 8/4/2007
Time: 20:31:16

Inventor Name Search Result

Your Search was:

Last Name = SINGHAL
First Name = VIVEK

Application#	Patent#	Status	Date Filed	Title	Inventor Name
09591986	7096418	150	06/12/2000	DYNAMIC WEB PAGE CACHE	SINGHAL, VIVEK
60179811	Not Issued	159	02/02/2000	Dynamic web page cache	SINGHAL, VIVEK
60201166	Not Issued	159	05/02/2000	Dynamic web page cache	SINGHAL, VIVEK
09781910	6792436	150	02/09/2001	METHOD FOR SYNCHRONIZING MULTIPLE SOFTWARE CACHES IN A MEMORY	SINGHAL, VIVEK P.
10679015	Not Issued	71	10/02/2003	High availability via data services	SINGHAL, VIVEK P.
60181664	Not Issued	159	02/11/2000	Method for synchronizing multiple software caches in a memory	SINGHAL, VIVEK P.

Inventor Search Completed: No Records to Display.

Search Another: Inventor

Last Name	First Name	
SINGHAL	VIVEK	Search

To go back use Back button on your browser toolbar.

Back to PALM | ASSIGNMENT | OASIS | Home page

DERWENT-ACC-NO: 2004-666264

DERWENT-WEEK: 200465

TITLE: Method for synchronizing cache objects in distributed cache management system, involves transmitting changed version of data object to management systems, without locking central database during database write operation

INVENTOR: PICOLET, R D; SINGHAL, V P ; ZHU, R

PATENT-ASSIGNEE: PERSISTENCE SOFTWARE INC[PERSN]

PRIORITY-DATA: 2000US-181664P (February 11, 2000) , 2001US-0781910 (February 9, 2001)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 6792436 B1	September 14, 2004	N/A	016	G06F 017/30

APPLICATION-DATA:

PUB-NO	APPL-DESCRIPTOR	APPL-NO	APPL-DATE
US 6792436B1	Provisional	2000US-181664P	February 11, 2000
US 6792436B1	N/A	2001US-0781910	February 9, 2001

INT-CL (IPC): G06F017/30

ABSTRACTED-PUB-NO: US 6792436B

BASIC-ABSTRACT:

NOVELTY - The method involves providing a control attribute for each data object in a central database (15). When received information of the object is valid with respect to the attribute, a changed version of the object

is
established. The changed version of the object is transmitted to management
systems, without locking the central database during the database write
operation.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(1) apparatus for synchronizing cache object in distributed cache management
system;

(2) apparatus for synchronizing transaction in local cache management system;
and

(3) program storage device storing program for synchronizing cache objects in
distributed cache management system.

USE - For synchronizing cache objects in distributed cache management system
(DCMS).

ADVANTAGE - Communicates object state information among caches, without need
for verification through the central database. Hence even if messages are lost
or received out of order, synchronization of the data objects is performed
efficiently.

DESCRIPTION OF DRAWING(S) - DESCRIPTION OF DRAWING - The figure shows the block
diagram of the information management system.

central database 15

computer systems 25-27

processor 30

memory 31

CHOSEN-DRAWING: Dwg.1/4

TITLE-TERMS: METHOD SYNCHRONISATION CACHE OBJECT DISTRIBUTE CACHE MANAGEMENT

SYSTEM TRANSMIT CHANGE VERSION DATA OBJECT MANAGEMENT SYSTEM LOCK
CENTRAL DATABASE DATABASE WRITING OPERATE

DERWENT-CLASS: T01

EPI-CODES: T01-F02C1; T01-H03A; T01-S03;

SECONDARY-ACC-NO:
Non-CPI Secondary Accession Numbers: N2004-527519

DERWENT-ACC-NO: 2006-645739

DERWENT-WEEK: 200667

TITLE: Dynamic web pages caching method involves updating cache by refreshing or deleting data page, if received event changes page dependency data

INVENTOR: EMMONS, I; JENSEN, R ; SINGHAL, V

PATENT-ASSIGNEE: PERSISTENCE SOFTWARE INC[PERSN]

PRIORITY-DATA: 2000US-0591986 (June 12, 2000) , 2000US-179811P (February 2, 2000) , 2000US-201166P (May 2, 2000)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 7096418 B1	August 22, 2006	N/A	029	G06F 017/00

APPLICATION-DATA:

PUB-NO	APPL-DESCRIPTOR	APPL-NO	APPL-DATE
US 7096418B1	Provisional	2000US-179811P	February 2, 2000
US 7096418B1	Provisional	2000US-201166P	May 2, 2000
US 7096418B1	N/A	2000US-0591986	June 12, 2000

INT-CL (IPC): G06F017/00

ABSTRACTED-PUB-NO: US 7096418B

BASIC-ABSTRACT:

NOVELTY - A data page including page dependency indicating an underlying data source on which the data page is dependent, generated by a Request-Based dependency generator, is received from a server. The data page and page

dependency data are stored. The cache is updated by refreshing or deleting the
data page, if a received event changes one of the page dependency data.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(1) computer software product storing instructions for caching dynamic web
pages; and

(2) proxy server system.

USE - For caching dynamic web pages.

ADVANTAGE - Dynamic web pages can be stored in a cache, and refreshed efficiently, to timely respond to requests for page content, so that workload
on internet servers, is reduced, while enabling the user to retrieve valid web
pages.

DESCRIPTION OF DRAWING(S) - The figure shows a block diagram of dynamic web
pages caching system.

dynamic web pages caching system 200,201 2A, 2B/11

TITLE-TERMS: DYNAMIC WEB PAGE METHOD UPDATE CACHE REFRESH DELETE DATA PAGE
RECEIVE EVENT CHANGE PAGE DEPEND DATA

DERWENT-CLASS: T01

EPI-CODES: T01-H01C3; T01-H03A; T01-N01D4; T01-S03;

SECONDARY-ACC-NO:
Non-CPI Secondary Accession Numbers: N2006-520417

Web Results **1 - 10** of about **106,000** for **commit transaction cache synchronization message performed asynchronousl**

Did you mean: commit transaction ***cachesynchronization*** message performed asynchronously

Persistent **cache synchronization** and start up system - US Patent ...
Asynchronous resynchronization of a **commit** procedure .... A method of persistent **cache synchronization** for communications over an external communication ...
www.patentstorm.us/patents/6061714-claims.html - 69k - Cached - Similar pages

> Persistent **cache synchronization** and start up system - US Patent ...
> **Asynchronous** resynchronization of a **commit** procedure .... The second **cache** then transmits a checkpoint confirmation **message** to the first computer responsive ...
> www.patentstorm.us/patents/6061714-description.html - 106k - Cached - Similar pages
> [More results from www.patentstorm.us]

OSDI '06 Paper
Asynchronous file systems provide explicit **synchronization** operations such ..... The total number of **transactions performed** by TPC-C is approximately twice ...
www.usenix.org/events/osdi06/tech/nightingale/nightingale_html/index.html - 88k - Cached - Similar pages

[PS] 1 Introduction
File Format: Adobe PostScript - View as Text
As **transactions commit**, modications are written to the log and also inserted ... **performed asynchronously** by a background thread that moves modied objects ...
research.microsoft.com/~adya/pubs/sigops96.ps.gz - Similar pages

ProdAgentLiteMessageService < CMS < TWiki
The operation **commit** closes the current **transaction**, making all **message** operations to ... The **message** service on the other side, is an **asynchronous message** ...
https://twiki.cern.ch/twiki/bin/view/CMS/ProdAgentLite**Message**Service - 44k - Cached - Similar pages

Asynchronous coordinated **commit** replication and dual write with ...
As the succession of **transaction** steps or operations are **performed**, ... A prepare-to-**commit message** is sent and is followed by a response (phase 1). ...
www.freepatentsonline.com/7177866.html - 193k - Cached - Similar pages

> **Asynchronous** coordinated **commit** replication and dual write with ...
> As the succession of **transaction** steps or operations are **performed**, ... When this has committed successfully, a **commit message** is sent to the target node ...
> www.freepatentsonline.com/20040133591.html - 182k - Cached - Similar pages

[PDF] **Synchronization** and recovery in a client-server storage system
File Format: PDF/Adobe Acrobat
message. to the server. 2. Send **asynchronously** to the private log all remaining log records. 3. Send a. **commit transaction**. **message** to the server and wait ...
portal.acm.org/ft_gateway.cfm?id=765557&type=pdf&dl=GUIDE&dl=ACM - Similar pages

[PDF] Persistent Object **Synchronization** with Active Relational Databases
File Format: PDF/Adobe Acrobat
This attribute will be used, during the RDBMS **transaction commit**, ... To initiate the **synchronization messages**, the CMC object invokes the ...
ieeexplore.ieee.org/iel5/6380/17059/00787606.pdf?tp=&arnumber=787606&isnumber=17059 - Similar pages

[PDF] OCRed document
File Format: PDF/Adobe Acrobat
When the user decides to **commit** the **transaction**, it invokes the corresponding .... To initiate the **synchronization messages**, the CMC object invokes the ...
ieeexplore.ieee.org/iel5/6380/17059/00787606.pdf?arnumber=787606 - Similar pages

Did you mean to search for: commit transaction ***cachesynchronization*** message performed asynchronously

1 2 3 4 5 6 7 8 9 10 **Next**

Google

cache synchronization message performed as | Search | Advanced Search Preferences

New! View and manage your web history

Web Results **1 - 10** of about **395,000** for **cache synchronization message performed asynchronously**. (**0.15** seconds)

Tip: Save time by hitting the return key instead of clicking on "search"

Background **cache synchronization** - US Patent 6553409
Background **cache synchronization** - US Patent 6553409 from Patent Storm. ... System for **asynchronously** delivering enqueue and dequeue information in a pipe ...
www.patentstorm.us/patents/6553409.html - 22k - Cached - Similar pages

Flushing of **cache** memory in a computer system - US Patent 5893149
an **asynchronous** flusher configured to flush all identified data lines in said selected page in said **cache** memory of said requesting subsystem, wherein said ...
www.patentstorm.us/patents/5893149-claims.html - 26k - Cached - Similar pages

The <session> Element
<is-**asynchronous**>, Specifies whether **cache synchronization** is **performed asynchronously** .... Specifies whether the session logs uncaught exception **messages**. ...
www.oracle.com/.../state/content/navId.4/navSetId._/vtAnchor.655917/vtTopicFile.tl_mappings%7Ctrouble7~html/ - 61k - Cached - Similar pages

Olston, Chris; Widom, Jennifer: Best-Effort **Cache Synchronization** ...
The resources for **cache synchronization** may be limited at a number of points. .... and instead resort to **asynchronous** propagation of all database updates, ...
dbpubs.stanford.edu/pub/2002-14 - 71k - Cached - Similar pages

[PDF] An **Asynchronous** Protocol for Release Consistent Distributed Shared ...
File Format: PDF/Adobe Acrobat
performed asynchronously. In our protocol, we separate the ..... for transferring **synchronization messages** and the other is a. high speed ethernet for ...
portal.acm.org/ft_gateway.cfm?id=338550&type=pdf - Similar pages

[PDF] An **Asynchronous** Protocol for Release Consistent Distributed Shared ...
File Format: PDF/Adobe Acrobat
are **performed asynchronously**. In the proposed protocol, ... so that the **synchronization messages** can be immediately sent and delivered even if ...
www.springerlink.com/index/W5716133863R45T5.pdf - Similar pages

[PS] Best-Effort **Cache Synchronization** with Source Cooperation
File Format: Adobe PostScript - View as Text
as they occur and participating in **cache synchronization** with any. spare bandwidth. .... and instead resort to **asynchronous** propagation of all database up- ...
infolab.stanford.edu/~olston/publications/bes-sigmod.ps - Similar pages

[PS] An **Asynchronous** Protocol for Release Consistent Distributed Shared ...
File Format: Adobe PostScript - View as Text
information are **performed asynchronously**. In our protocol, we separate the ..... one is a 10 MB Ethernet used for transferring **synchronization messages** and ...
dcslab.snu.ac.kr/paper/sac2k-arc.ps - Similar pages

[PS] An **Asynchronous** Protocol for Release Consistent Distributed Shared ...
File Format: Adobe PostScript - View as Text
performed asynchronously. In the proposed protocol, the communication is ..... one is a 10 MB Ethernet used for transferring **synchronization messages** and ...
dcslab.snu.ac.kr/paper/JSC0102-arc.ps - Similar pages

INF: Understanding Bufwait and Writelog Timeout **Messages**
SQL Server uses normal thread **synchronization** methods to guarantee ordered access to the transaction log. In a writelog timeout **message**, the database ID is ...
support.microsoft.com/kb/167711 - Similar pages

1 2 3 4 5 6 7 8 9 10 **Next**

---

cache synchronization message per Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

---

United States Patent 6553409

Background cache synchronization

US Patent Issued on April 22, 2003

ABSTRACT | CLAIMS | DESCRIPTION | FULL TEXT

Inventor(s)

Rajeev Dujari
Danpo Zhang
Josh Cohen
E. Castedo Ellerman

Assignee

Microsoft Corporation

Application

No. 350331 filed on 1999-07-09

Current US Class

709/213 , 709/214 , 711/147

Field of Search

345/541 , 709/213 , 709/214 , 709/231 , 709/235 , 709/246 , 711/118 , 711/141 , 711/144 , 711/147 , 711/154 , 725/131

Examiners

Primary: David Wiley
Assistant: George Neurauter

Attorney, Agent or Firm

Law Offices of Albert S. Michalik, PLLC

US Patent References

5390318
5442760
5448708
5517652
5617537
5628015
5737599
5787470
5787475

Abstract

Two new cache control headers in the cache control header, "post-check" and "pre-check", enable the display of content from the cache, with a later synchronization of the content performed in the background via a conditional request such as an IMS request. These headers enable the server to define a non-validate time period relative to the cached content's age in which the user will receive content from the cache, a background synchronization period in which the user will receive content from the cache and automatically queue a request for background synchronization thereof, and a validate period in which the cached content may or may not be used, depending on a response to a validation request sent to the server. The content is quickly rendered for the user in the non-validate and background synchronization periods, and the hit count is correct in the background synchronization and validate periods. In the background synchronization time period, the user has a fast experience with rapidly rendered content, while via the background synchronization, the server receives the proper number of hits.

5829022
5889958
5906658
5923855
5987506
5991760
6026474
6061715
6067565
6085226
6112231
6119153
6219728
6233606

United States Patent 5893149

Flushing of cache memory in a computer system

US Patent Issued on April 6, 1999

Inventor(s)

Erik E. Hagersten

Aleksandr Guzovskiy

Assignee

Sun Microsystems, Inc.

Application

No. 673881 filed on 1996-07-01

Current US Class

711/135, 711/113, 711/118, 711/119, 711/124, 711/141, 711/159

Examiners

Primary: Tod R. Swann

Assistant: Fred F. Tzeng

Attorney, Agent or Firm

Conley, Rose & Tayon, P.C.; Kivlin; B. Noel

US Patent References

5025365
5303362
5313609
5408636
5611070
5634068

ABSTRACT | CLAIMS | DESCRIPTION | FULL TEXT

Claims

What is claimed is:

1. A method for replacing data while maintaining coherency of said data within a computer system having at least a first subsystem and a second subsystem coupled to each other via an interconnect, wherein said first subsystem and said second subsystem each include a local interconnect, a global interface, at least one processor, and at least one cache, and wherein said first subsystem is a home of said data, the method comprising:

selecting a page of data in said at least one cache of said second subsystem, wherein said page of data contains a plurality of data lines;

freezing all accesses of said second subsystem to said page of data;

identifying at least one data line of said plurality of data lines of said page of data, wherein said at least one data line contains data that requires flushing;

flushing said at least one data line of said plurality of data lines of said page of data, wherein said flushing includes issuing of a local flush instruction and a global flush instruction, said local flush instruction is issued by said at least one processor of said second subsystem to said global interface of said second subsystem, and said global flush instruction is issued by said at least one processor of said second subsystem to said first subsystem in response to said local flush instruction, and wherein said global flush instruction is performed asynchronously from said local flush instruction; and

maintaining a coherency between said at least one data line of said plurality of data lines of said page of data flushed in said second subsystem and data in said first subsystem.

2. The method as recited in claim 1 wherein said selecting a page of data in said at least one cache of said second subsystem includes a criterion suitable to select said page of data that is suitable for replacement.

3. The method as recited in claim 2 wherein said criterion includes a least recently used criterion.

4. The method as recited in claim 1 wherein said freezing all accesses of said second subsystem to said page of data includes denying local accesses initiated by said at least one processor of said second subsystem.

5. The method as recited in claim 1 wherein said identifying at least one data line of said plurality of data lines of said page of data includes determining if said data line is in valid status.

6. The method as recited in claim 5 wherein said valid status of said data line includes either an owned state or a modified state, wherein said owned said indicates that said second subsystem has a cached copy of said data line and said modified state indicates that said second subsystem is the sole owner of said data line.

7. The method as recited in claim 6 wherein a subsystem that has said owned state of said data line is configured to perform a write-back upon replacement of said data line.

8. The method as recited in claim 1 wherein said flushing said at least one data line of said plurality of data lines of said page of data is executed when said data line is in said valid status, and said flushing is not executed if said data line is in an invalid status, wherein said invalid status indicates that said second subsystem can discard said data line.

9. The method as recited in claim 8 wherein said invalid status of said data line includes either an invalid state or a shared state, wherein said invalid state indicates that said second subsystem has no cached copy of said data line, and said shared state indicates that said second subsystem has a shared cached copy of said data line.

10. The method as recited in claim 1 wherein said maintaining said coherency comprises:

sending a synchronization request from said second subsystem to said first subsystem;

verifying that said data is now coherent between said first subsystem and said second subsystem; and

sending an acknowledgment from said first subsystem to said second subsystem in response to said synchronization request, said acknowledgment indicating that said data is now coherent between said second and said first subsystem.

11. The method as recited in claim 10 wherein said acknowledgement by said first system to said second subsystem comprises either a first message, a second message, or a third message; wherein said first message is indicative of either said owned state or said modified state of said data line, said second message is indicative of said shared cached copy of said data line, and said third message is indicative of said no cached copy of said data line.

12. A method for flushing a selected page of data in a cache memory in a requesting subsystem while maintaining coherency of said data with a home subsystem within a computer system, wherein said selected page of data comprises a plurality of data lines, said requesting subsystem is coupled to said home subsystem via an interconnect, said requesting subsystem and said home subsystem each include a local interconnect, a global interface, at least one processor, and at least one said cache memory, wherein said computer system comprises at least said home subsystem and requesting subsystem, the method comprising:

issuing a local flush instruction for each data line of said plurality of data lines within said selected page in said cache memory of said requesting subsystem by said at least one processor, wherein said data line is identified for replacement;

issuing a global flush instruction by said at least one processor of said requesting subsystem to said home subsystem in response to said local flush instruction, wherein said issuing of said global flush instruction is performed asynchronously from said local flush instruction;

sending an acknowledgment message by said home subsystem to said requesting subsystem in response to said global flush instruction issued by said requesting subsystem, wherein said acknowledgement message is indicative of a status of said data line;

issuing of a read-to-own transaction by said requesting subsystem on said local interconnect of said requesting subsystem in response to said acknowledgment message indicative of at least one of said status of said data line;

sending a completion message by said requesting subsystem to said home subsystem; and

updating a directory in said home subsystem by said home subsystem in response to said completion message, wherein said directory comprises at least one entry indicative of said status of said data line.

13. The method as recited in claim 12 wherein said status of said data line comprises one of either an owned state, a modified state, a shared state, or an invalid state of said data line, wherein said owned state indicates at least said home subsystem has a cached copy of said data line, said modified state indicates said home subsystem is the sole owner of said data line, said shared state indicates at least said home subsystem has a shared copy of said data line, and said invalid state indicates said home subsystem has no copy of said data line.

14. The method as recited in claim 13 wherein said sending said acknowledgment message by said home subsystem comprises either a first message, a second message, or a third message; wherein said first message is indicative of either said owned state or said modified state of said data line, said second message is indicative of said shared state of said data line, and said third message is indicative of said invalid state of said data line.

15. The method as recited in claim 14 wherein either said second message or said third message sent by said home subsystem to said requesting subsystem further indicates that said requesting subsystem can discard said data line.

16. The method as recited in claim 12 wherein said flushing is further comprises synchronization of all identified data lines in said page of data of said requesting subsystem wherein said synchronization results in every line of said all identified data lines having said invalid state in said home subsystem.

17. A system for replacing a selected page of data in a cache memory in a requesting subsystem while maintaining coherency of said data with a home subsystem within a computer system, wherein said selected page of data comprises a plurality of data lines, said requesting subsystem is coupled to said home subsystem via an interconnect, said requesting subsystem and home subsystem each includes a local interconnect, a global interface, at least one processor, and at least one said cache memory, wherein said computer system comprises at least said home subsystem and requesting subsystem, the mechanism comprises:

a detector configured to detect a need to replace said data in said cache memory of said requesting subsystem;

an identifier configured to identify said selected page of data in said cache memory and to identify at least one data line of said plurality of data lines in said selected page of data in said requesting subsystem;

an asynchronous flusher configured to flush all identified data lines in said selected page in said cache memory of said requesting subsystem, wherein said asynchronous flusher is further configured to:

issue a local flush instruction for each identified data line of said plurality of data lines,

issue a global flush instruction to said home subsystem in response to said local flush instruction, wherein said global flush instruction is asynchronous to said local flush instruction,

receive an acknowledgment message sent by said home subsystem in response to said global flush instruction, wherein said acknowledgement message is indicative of a status of said identified data line,

issue a read-to-own transaction on said local interconnect of said requesting subsystem in response to said acknowledgment message, and

send a completion message to said home subsystem.

18. The system as recited in claim 17 said home subsystem updates a directory in said home subsystem in response to said completion message, wherein said directory comprises at least one entry indicative of said status of said identified data line.

19. The system as recited in claim 17 wherein said status of said identified data line comprises one of either an owned state, a modified state, a shared state, or an invalid state of said identified data line, wherein said owned state indicates at least said home subsystem has a cached copy of said identified data line, said modified state indicates said home subsystem is the sole owner of said identified data line, said shared state indicates at least said home subsystem has a shared copy of said identified data line, and said invalid state indicates said home subsystem has no copy of said identified data line.

20. The system as recited in claim 17 wherein said receive said acknowledgment message sent by said home subsystem message comprises either a first message, a second message, or a third message; wherein said first message is indicative of either said owned state or said modified state of said selected data line, said second message is indicative of said shared state of said selected data line, and said third message is indicative of said invalid state of said selected data line, and wherein either said second message or said third message further indicates that said requesting subsystem can discard said identified data line.

21. The system as recited in claim 17 further configured to synchronize said flush of said selected page of data of said cache memory of said requesting subsystem until every said identified data line of said selected page of data has said invalid state in said home subsystem.


.NET 2.0 for Delphi Programmers
Shemitz, Jon

United States Patent 6061714

Persistent cache synchronization and start up system

US Patent Issued on May 9, 2000

ABSTRACT CLAIMS DESCRIPTION FULL TEXT

Inventor(s)

Barron Cornelius Housel, III
Ian Beaumont Shields
Teresa Anne Meriwether

Assignee

International Business Machines Corporation

Application

No. 852257 filed on 1997-05-07

Current US Class

709/203

Field of Search

709/200, 709/201, 709/203, 709/217, 709/218, 709/219

Examiners

Primary: Moustafa M Meky

Attorney, Agent or Firm

Ray-Yarletts; Jeanine S. Myers Bigel Sibley & Sajovec

US Patent References

5319773
5319774
5371886
5426645
5428771
5432926
5446904
5469503
5500890
5539736
5546582

Abstract

Method, appartus and program products for persistent cache synchronization for communications over an external communication link between a client protocol conversion application executing on a first computer and a server protocol conversion application executing on a second computer located remote from the first computer are provided. Protocol conversion applications are provided on both the host side and the terminal side of the external communication link to provide communications over the external communication link using a differenced communication data stream which includes a reduced volume of data for transmittal. A checkpoint is provided to allow for improved performance on start up of a new communication session over the external communication link between the client application and the server application by initiating a checkpoint request and transmitting the request from the client to the server. The active protocol cache of the server is copied to a checkpoint cache responsive to a received checkpoint request. A checkpoint confirmation message is sent to the client from the server. Responsive to receipt of a checkpoint confirmation message indicating successful copying of the server active protocol cache to a checkpoint cache, the client application creates a

5551043
5561797
5581704
5581753
5592512
5594910
5611038
5613060
5666399
5682514
5706435
5724581
5734898
5751719
5754774
5758072
5758174
5765004
5768538
5781908
5787470
5802267
5813032
5832508
5907678

checkpoint cache of the first computer as a copy of the protocol cache of the first computer. The first computer creates a checkpoint cache by first creating a temporary cache before initiating the checkpoint and on receipt of a confirmation from the server, converting the temporary cache to a confirmed checkpoint cache.

Foreign Patent References

1322607 CA Jan., 1989
61-41063 JP Feb., 1986
95/10805 WO Apr., 1995
97/46939 WO Dec., 1997

Other References

Application to Host File Transfer Restart Method, IBM Technical Disclosure Bulletin, vol. 31, No. 5, pp. 409-410 (Oct. 1988).

Synchronous Interleaved I/O File Server, IBM Technical Disclosure Bulletin, vol. 32, No. 9B, pp. 91-92 (Feb. 1990).

Client/Server-based File Transmission Checkpoint/Restart Protocol, IBM Technical Disclosure Bulletin, vol. 38, No. 09, pp. 191-193 (Sep. 1995).

Two-Phase Commit Resynchronization, IBM Technical Disclosure Bulletin, vol. 39, No. 01, pp. 79-80 (Jan. 1996).

Jacob Ziv, Abraham Lempel; Compression of Individual Sequences via Variable-Rate Coding, IEEE Transactions on Information Theory, vol. I-24, No. 5, Sep. 1978.

Combining Presumed Abort Two-Phase Commit Protocols with SNA's Last Agent Optimization; IBM Technical Disclosure Bulletin, vol. 34, No. 7B, pp. 334-338 (Dec. 1991).

Efficient Commit Protocol for Shared Nothing Architectures Using Common Log Server, IBM Technical Disclosure Bulletin, vol. 36, No. 12, pp. 65-66 (Dec. 1993).

Emulation Data Stream, IBM Technical Disclosure Bulletin, vol. 33, Aug. 1990, pp. 221-223.

Jacob Ziv, Abraham Lempel; A Universal Algorithm Sequential Data Compression, IEEE Transactions on Information Theory, p. 337-343, May 1977.